Lab 3: Java RMI

1. Introduction

In the previous lessons, we have worked our way over sockets. We now are able to build web applications which can interactive web applications. More precisely, we can build the "outside view" of such an application, we can build the interface with which the user interacts over the web.

But there is also the "inside view", the things under the hood. Real applications in enterprise environments are quite complex and often are interconnected with other applications: The software of the human resources department of a company and the software of the financial department of that company need to work together. Changes in the status of the employees need to automatically be reflected in their salary payments. The financial department's software will also need to interact with the software of banks to automatically issue salary payments. It becomes clear that this sort of connectivity and automation cannot be achieved with JSPs or plain servlets alone. Some "glue" technology is needed.

The glue technology is "remote procedure calls" (RPCs), the ability that a process on my computer can tell a process on another computer to invoke a procedure (such as "pay salary to account xyz"). The code of that procedure will be unknown to my computer and the execution takes place entirely on the other system.

In this lecture, we will learn how to perform Java Remote Method Invocation.

2. Examples

2.1. RemotePrintServer / RemotePrintClient

The interface RemotePrintInterface provides a method `print` which receives one `String`. The RMI Server RemotePrintServer implements this method in a straightforward fashion: it prints the `String` to `System.out`. An instance of this server is created and registerd as RMI server object under the name `server` in a registry listening for requests on port `9999`.
The RemotePrintClient wants to make use of this very valuable print service: It first obtains the registry and then, from this registry, the object names `server`. This object must be an instance of RemotePrintInterface, so we can call its `print` method. When doing so, the string we pass in is sent to the server object. Since the object actually resides in a different Java virtual machine in a different process, the string is marshalled and sent over a socket to that other process, read and unmarshalled from the socket, handed to the server object, and then printed. This would work over the network as well, we can call a method which will be executed on a different computer. And we can do so relatively conveniently without bothering too much about details.

1. RemotePrintInterface.java
2. RemotePrintServer.java
3. RemotePrintClient.java

2.2. RemotePrintServer / RemotePrintClientErroneous

As the name RemotePrintClientErroneous implies, this is a "wrong" version of the above example. The error is located in an incorrect typecast in the RemotePrintClientErroneous program: The program incorrectly assumes that the object returned from the registry can be cast to RemotePrintServer, which it cannot. It can only be cast to RemotePrintInterface, as done in RemotePrintClient.

But why is that? The reason is already mentioned in the text of the first example: The actual instance of RemotePrintServer resides in a different virtual machine (process) from the client's process. Its actual implementation code is unknown to the client. The object returned by the registry to the client is just a Proxy instance which "implements" all interfaces of the original object which inherit from Remote. RemotePrintInterface is such an interface. If a method of the proxy object is called, it will send a message with the method name and parameters to the actual object on the server side. The server object will invoke the method and send back the results. Then the proxy's method can return as well. This way, the communication and "remoteness" of the procedure invocation is invisible (transparent) to the user of the client object. And since the proxy object is not the real object, we cannot cast it to RemotePrintServer without receiving an exception.

1. RemotePrintInterface.java
2. RemotePrintServer.java
3. RemotePrintClientErroneous.java